TECNYCE

NYCE

Course Outline

7 www.nyce.org.mx

Lesson 1

- Introduction
- Android Software Layers
- Android Libraries
- Components of an Android application
- Application Life-cycle
- Pre-requisites for Android Application Development
- Android Studio
- Gradle
- Lab 1: Creating Your First Application

8 www.nyce.org.mx

TECNYCE

NYCE

Lesson 2

- The Android Manifest File
- Structure of the Manifest File
- Android SDK Tools
- Activity life-cycle through Java
- Create an Activity
- Fragments
  - Fragments life-cycle
  - Creating a Fragment Sub-Class
- Lab 2: Controlling a Camera's Flash Light

9 www.nyce.org.mx

Lesson 3

- Introduction
- Fragments
  - Fragments life-cycle
  - Creating a Fragment Sub-Class
- Views
  - Using Views
  - Adding a View to your application
- List Views and List Activity
  - Using a ListActivity
  - Adding ListView
  - Add Event to List Items
- Lab 3: Creating a Simple To-Do List Application

10 www.nyce.org.mx

Lesson 4

- Introduction
- Intents
  - Explicit Intents
  - Implicit Intents
- Native Android Actions
- Data Transfer
- Intent to Call Activities
  - Direct calls
  - Sub-activities: Calling Activities for Results
- Register an IntentFilter
- Lab 4: Creating Contacts Selection Application

Lesson 5

- Introduction
- Views
- Layouts
  - Layout Properties
  - Loading the Layout from Code
  - Creating and Editing Layouts in Android Studio
- Customized Views
- Modify Existing Views
  - Step by Step
  - What is in onDraw()?
- Lab 5: Custom View, Drawer Layout, and Fragments Application

Lesson 6

- Introduction
- Android Resources
  - Why Using Resources?
  - Adding Resources to your Application
  - Using Resources
  - Types of Resources
- Android Themes and Styles
  - Creating Themes
- Android Material Design
  - Using the Material Theme
  - Color Palette
- Lab 6: A To-Do List Application in Material Design

Lesson 7

- Introduction
- User Interaction through Messages
- Dialogs
  - Dialog sub-classes
  - Creating dialogs with user-defined layout
  - Creating an Alert Dialog
  - Creating a Progress Dialog
  - Activities with Dialog Theme
- Toasts
- Menus
  - Sub-menus
  - Context Menus
  - Popup Menus
- WebView
- Lab 7: Wallpaper Application

Lesson 8

- Android Storage Options
- File I/O
- Shared Preferences
  - Retrieving Shared Preferences
  - Save activity state
- Connecting to the internet
- Background Processing
  - Java Threads and AsyncTask
  - Android Services
  - Background Fragments
- Lab 8: Quotes Provider Application

Lesson 9

- Introduction
- SQLite Database in your application
- SQLite library
- SQLiteDatabase
- Cursors
- Databases in Android
- Content Providers
  - Native Android Content Providers
  - Custom Content Provider
  - Designing the Storage for Data
  - Steps to Create a Content Provider
- Sync Adapters
- Lab 9: SQLite Databases and Content Providers

Lesson 10

- Introduction
- Creating a notification
- Notification actions
- Steps to Create a Simple Notification
- Expandable Notifications
- Notifications Layouts
- Notification Priority
- Notifications in Android 5.0 (Lollipop)
  - Lock Screen Notifications
  - Heads-up Notifications
- Lab 10: Implementing Android Notifications

Android Application Development

LESSON 1

ANDROID FRAMEWORK AND ANDROID STUDIO

ATC

AND-401

Outline

- Introduction
- Android Software Layers
- Android Libraries
- Components of an Android application
- Application Life-cycle
- Pre-requisites for Android Application Development
- Android Studio
- Gradle
- Lab 1: Creating Your First Application

19 www.nyce.org.mx

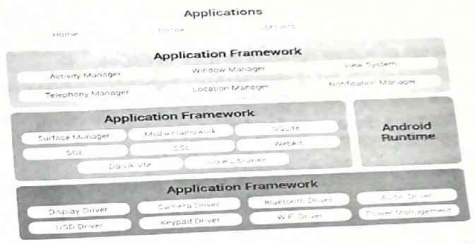
Introduction

- Android software development kit (SDK), Android libraries and its application framework and life-cycle.
- All labs will develop applications using Android Studio – the official Android integrated development environment (IDE).
- Labs will also target Android 5.0 (Lollipop).

Android Studio

20 www.nyce.org.mx

Android Software Layers



21 www.nyce.org.mx

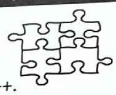
Linux kernel

- Used by Android for core system services:
  - Security, memory management, process management...etc.
- It acts as an abstraction layer between the hardware and the rest of the software st
- It has a proven driver model ar resource management.

Linux

22 www.nyce.org.mx

C/C++ Libraries



- A set of native libraries written in C/C++.
- Responsible for stable performance of various components.
- Manage the access for different processes to compose 2D and 3D graphic layers.
- Responsible for media framework processing

23 www.nyce.org.mx

Android Runtime (ART)

- The managed runtime used by applications.
- The successor of the Dalvik runtime.
- ART executes the Dalvik Executable format and Dex bytecode specification - byte-code optimized files that are more efficient to run on small processes.
- Android runtime contains core libraries that are written in Java programming language and contains a collection of classes, utilities, I/O and other tools.

24 www.nyce.org.mx

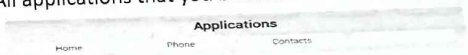
Application Framework

- Written in Java programming language.
- A collection of Java libraries that all Android applications use.
- Has several components. For example:
  - ✓ Activity Manager manages the life cycle of the applications
  - ✓ Package Manager keeps track of applications installed on the device
  - ✓ Telephony Manager contains a set of APIs necessary for calling applications.

25 www.nyce.org.mx

Application Layer

- The upper most layer of the Android Architecture
- Contains all the applications used by the end user.
- All applications that you build are located in this layer.



26 www.nyce.org.mx

Android Libraries

- A set of Java APIs for developing your applications. All Android devices must provide support for these libraries
  - android.view , android.widget, android.util, android.database, android.content, android.app, android.provider, android.telephony, android.webkit
- Also contains a set of advanced Android libraries:
  - android.location, android.media, android.opengl, android.hardware, android.bluetooth...etc.

27 www.nyce.org.mx

Components of an Android application

- ***Activities***

  The application's user interface. Every screen in an application extends the Activity class.
- ***Services***

  The invisible part of your application that runs in the background.
- ***Content Providers***

  A data store that is exposed for sharing.
- ***Broadcast Receivers***

  Consume messages broadcasted by Intents.

28 www.nyce.org.mx

Components of an Android application

- **Views**

  Objects that know how to draw itself to the screen.
- **Intents**

  Objects used to send message across the whole Android system.
- **Notifications**

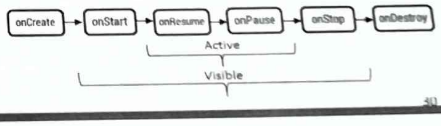
  Used in an application to alert users to certain events without having a visible activity.

29

Application Life-cycle

- Android applications' life cycle is controlled by the Android framework itself.
- Each Android application runs its own process and process management is handled exclusively at run time.

Activity Life Cycle



30

Type of Android processes and their priorities

- *Foreground (Active) processes*
- *Visible process*
- *Services*
- *Background process*
- *Empty process*

Android Studio

- The official IDE for Android application development.
- Provides a new and rich layout editor, with support for drag-and-drop theme editing.
- Building application is based on Gradle build system.
- Builds multiple APKs for various application build types.
- A new set of tools to support to resolve performance, usability and other issues.

Pre-requisites for Android Application

- *Android SDK platform.*
- *Integrated Development Environment (IDE)*
- *Android SDK Tools.*
- *Android SDK build tools.*

Gradle

- A build automation system.
- Combines the power and flexibility of Ant scripting with the library and dependency management capabilities.
- Using Gradle in Android development allows you to:
  - Customize the build process.
  - Create multiple APKs with different features each.
  - Reuse resources across different project modules.

Curso Desarrollo de aplicaciones con Android
Material elaborado para NYCE, queda prohibida la reproducción parcial o total.

TECNYCE

NYCE

Android Application Development

Lesson 2

Android SDK Tools and Activity Class

ATC

AND-401

www.nyce.org.mx

TECNYCE

NYCE

The Android Manifest File

- The most important in every Android applications.
- Must be located under the root directory of the application package.
- Should be named **AndroidManifest.xml.**
- An XML file that provides essential information about the application to the Android system.

37 www.nyce.org.mx

Outline

- *The Android Manifest File*
- *Structure of the Manifest File*
- *Android SDK Tools*
- *Activity life-cycle through Java*
- *Create an Activity*
- *Lab 2: Controlling a Camera's Flash Light*

36 www.nyce.org.mx

The Android Manifest File

- Includes:
  - ✓ The Java package names of the application, which is a unique identifier for the application.
  - ✓ A detailed description of the application's components, i.e. the activities, services, broadcast receivers, and content providers. Each of these components has a special XML tag syntax.
  - ✓ The permissions that the application must have to access protected parts of the Android System.
  - ✓ The minimum level of the Android API that the application requires.

38 www.nyce.org.mx

Structure of the Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.lesson2"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.lesson2.MainActivity"
            android:label="@string/" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


Android SDK Tools

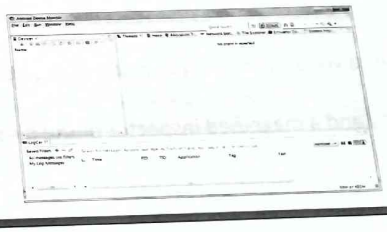
- Installed locally on your disk.
- Come with a set of tools that are very helpful for development, debugging, and maintainance.
- Located in folders: **<SDK>/android-sdk/platform-tools** and **<SDK>/android-sdk/tools** where **<SDK>** is the installation folder of your Android SDK.

Android SDK Tools

- **Android Device Monitor**
  - A stand-alone tool that is bundled with Android Studio.
  - Provides a graphical user interface (GUI) to access several Android debugging and analysis tools.
  - To access Android Device Monitor from Android Studio, click on its icon in the tool bar.

Android SDK Tools - DDMS

- Stands for Dalvik Debug Monitor Server.



Android SDK Tools - DDMS

- DDMS and Android Device Monitor are often used interchangeably
- A debugging tool which provides
  - Device screen capture
  - Thread and heap information on the device
  - LogCat
  - Process and radio state information
  - Incoming call and SMS spoofing
  - location data spoofing
  - ...etc

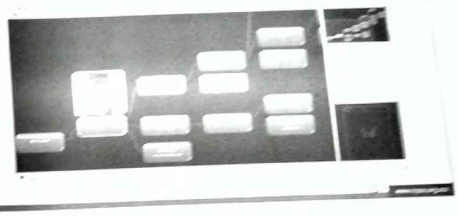
43

Android SDK Tools - HierarchyViewer

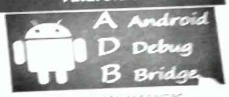
- Allows you to debug and optimize your application's user interface.
- Provides a visual representation of the layout's View hierarchy and a magnified inspector of the display.

44

Android SDK Tools - HierarchyViewer



Android SDK Tools



- Android Debug Bridge (ADB)
- A command line tool to communicate with an emulator instance or connected phone.
- A client-server program that includes:
  - A client that runs on your development machine
  - A server that runs on your development machine.
  - A daemon that runs on the emulator.

46

Android SDK Tools - ADB

- To use ADB:
  - Open a command prompt.
  - *Change directory to <SDK-installation-folder>/platform-tools.*
  - Type **adb** followed by a command.

47 www.nyce.org.mx

Android SDK Tools - LogCat

- A mechanism to collect and view system debug output.
- Logs from various applications and portions of the system are collected, then they can be viewed and filtered by the **logcat** command.
- You can use **logcat** from an ADB shell to view the log messages using command **adb logcat,** or using LogCat from within Android's Android Device Monitor.

48 www.nyce.org.mx

Android SDK Tools - LogCat

- Use **logcat** from an ADB shell by using command **adb logcat,** or using LogCat tool in Android's Android Device Monitor.

49 www.nyce.org.mx

Android SDK Tools - Battery Stats

- Available since Android 5.0 Lollipop
- It is part of the **dumpsys** tool and called **batterystats.**
- The tool generates useful statistics about the battery usage on a device for each application installed

50 www.nyce.org.mx

Building Backward-Compatible Android

- Latest Android 5.0 (Lollipop) provides a big number of new features and API methods.
- Building applications for the latest version, brings the challenges of using the many new features with older phones.
- Android Lollipop features would not be available in older devices.

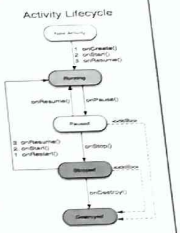
Building Backward-Compatible Android

- Android support libraries can provide backward-compatible versions of new Android API.
- The latest support library available is AppCompat v21, which provides the latest features of API level 21 to devices as old as API version 7.
- The following line in to file app/*gradle.build* adds the support library to your

```
dependencies {
    compile "com.android.support:appcompat-v7:21.0.+"
}
```


Activity life-cycle through Java

- Java callbacks are available for every state of an activity life-cycle.
- Life-cycle callbacks are functions called by the Android system whenever an event occurs.
- When you launch an Android application, Android system sends the callback Java method **onCreate()** which can be overriden.

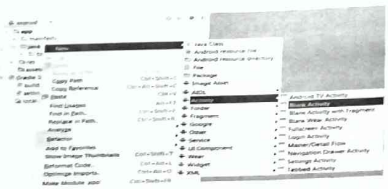


Activity life-cycle through Java

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
}
 @Override
public void onRestart(){
    super.onRestart();
}
 @Override
public void onStart(){
    super.onStart();
}
 @Override
public void onResume(){
    super.onResume();
}
```

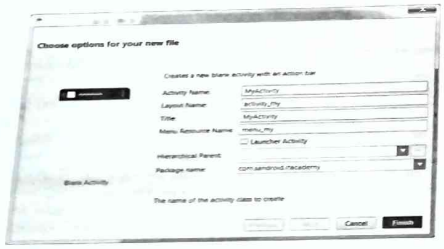
```
@Override
public void onPause(){
    super.onPause();
}
 @Override
public void onStop(){
    super.onStop();
}
 @Override
public void onDestroy(){
    super.onDestroy();
}
```

Create an Activity - 1



55 www.nyce.org.mx

Create an Activity - 2



56 www.nyce.org.mx

Create an Activity - 3

- Click on **Finish** and open the Java file created (*MyActivity.java*). Open the created class and notice that it extends the ActionBarActivity; which tells Android that the activity will have an Action Bar at the top.

```
public class MyActivity extends ActionBarActivity {
```

- Check the layout resource file *activity_my.xml* created under */res/layout*.

57 www.nyce.org.mx

Create an Activity - 4

- Each Activity must have a layout resource file that is inflated in the activity by using method `setContentView(R.layout.activity_my).`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
}
```

58 www.nyce.org.mx

Create an Activity - 5

- Activities created have a corresponding `<activity>` tag added to the *AndroidManifest.xml* file under */manfests/*.

```
<activity
    android:name="com.androidatc.MyActivity"
    android:label="@string/title_activity_my">
</activity>
```

59 www.nyce.org.mx

Android Application Development

Lesson 3

Fragments, Views, and List View

ATC

AND-401

www.nyce.org.mx

Outline

- Introduction
- Fragments
  - Fragments life-cycle
  - Creating a Fragment Sub-Class
  - Adding Fragments to a Layout
  - Adding Fragments Dynamically
- Views
  - Using Views
  - Adding a View to your application
- List Views and List Activity
  - Using a ListActivity
  - Adding ListView
  - Add Event to List Items
- Lab 3: Creating a Simple To-Do List Application

61 www.nyce.org.mx

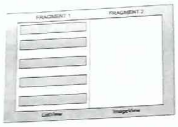
Introduction

- Views are essential components for user's interactions.
- Fragments are important to create complex and modular Activity UI.
- List views are compound Views that display a set of views vertically.

62 www.nyce.org.mx

Fragments

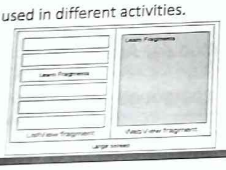
- Available since Android 3.0 (API level 11).
- They are a portion of the user interface in an Activity.
- They improve the UI reusability and allow the developer to create a multi-pane UI by:
  - Combining multiple fragments into a single activity.
  - Using the same fragment in multiple activities.



63 www.nyce.org.mx

Characteristics of Fragments

- You can add several fragments to an activity to build UI.
- Added and removed to an activity while during run-time.
- Have their own layout with their own lifecycle callbacks.
- Fragments can be used in different activities.



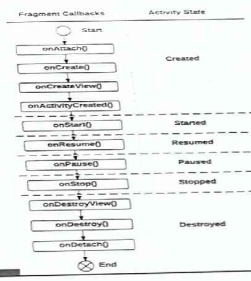
64 www.nyce.org.mx

Fragments life-cycle

- Linked to the life-cycle of the activity containing it.
- Can have three states just like an activity: Resumed, Paused, and Stopped.
- Similar to `onCreate()` method in Activity, method `onCreateView()` can be used to instantiate the user interface of a Fragment.
- Hosting activity manages the fragments stack by calling method `addToBackStack()`.

65 www.nyce.org.mx

Fragments



66 www.nyce.org.mx

Creating a Fragment Sub-Class

- Create a class that extends Fragment
- Override method onCreateView() and other lifecycle callback methods
- Some fragment callbacks:

  onCreateView(). Called when the system is drawing the layout of the fragment for the first time.

```
public class MyFragment extends Fragment {
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container,
      Bundle savedInstanceState) {
    // Set the fragment's UI
    return inflater.inflate(R.layout.my_fragment, container, false);
  }
}
```

67 www.nyce.org.mx

Creating a Fragment Sub-Class

- onCreate(). Invoked when creating the fragment
- onAttach(). Called when a fragment is first attached to its activity.
- onPause(). Called when the user is leaving the fragment.

68 www.nyce.org.mx

Adding Fragments to a Layout

```
activity_main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:tools="http://schemas.android.com/tools"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical">

<fragment
    android:name="com.androidatc.TopFragment"
    android:id="@+id/top_fragment"
    android:layout_weight="1"
    android:layout_height="match_parent" />

<fragment
    android:name="com.androidatc.BottomFragment"
    android:id="@+id/bottom_fragment"
    android:layout_weight="2"
    android:layout_height="match_parent" />

</LinearLayout>
```

69 www.nyce.org.mx

Adding Fragments Dynamically

- Get an instance of FragmentTransaction in your activity.

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

-

```
TopFragment fragment = new TopFragment();
fragmentTransaction.add(R.id.container, fragment);
fragmentTransaction.commit();
```

70 www.nyce.org.mx

Views

- View is a Java class that represents the basic building block for user interface components.
- It is the parent class of all other interactive UI components
- It occupies a rectangular area on the screen and is responsible for drawing and event handling.
- The **ViewGroup** subclass is the base class for *layouts*, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.

71 www.nyce.org.mx

Using Views

- Views in a window are arranged in a single tree.
- You can add views either from code or by specifying a set of views in XML layout files.
- ***Operations to set on Views:***
  - ***Set properties.***
  - ***Set focus.***
  - ***Set up listeners.***
  - ***Set visibility.***

72 www.nyce.org.mx

Adding a View to your application

1. Open a layout file .
2. Drag the **Button** component.

73 www.nyce.org.mx

Adding a View to your application

3. Save your work.
4. Run your application in the emulator to view your added button.

74 www.nyce.org.mx

List Views and List Activity

- You can manipulate list views in your application in one of two ways:
  - Using ListActivity - a sub type of Activity that is made up of a single List view only.
  - Adding a list view component to your own XML layout file.

75 www.nyce.org.mx

Using a ListActivity

- Create a String array of elements you want to display.

```
static final String[] ANDROID_OS = new String[] {"Cupcake", "Donut", "Eclaire", "Froyo",
"GingerBread", "HoneyComb", "Ice Cream Sandwich", "JellyBean" };
```

- Link your array of raw data to the list view using an

```
setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, ANDROID_OS));
```

76 www.nyce.org.mx

Using a ListActivity

77 www.nyce.org.mx

Adding ListView

Add a List View component to the layout.

78 www.nyce.org.mx

Adding ListView

- Get a reference to the ListView widget in Java.

```
public class ListFruitActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Get a reference to the list view widget
        ListView listView = (ListView) getElementById(R.id.listViewOS);
    }
}
```

- Link your array of raw data to the list view variable

```
listview.setAdapter(
        new ArrayAdapter<String>(this,R.layout.list_os, ANDROID_OS)
);
```


Add Event to List Items

To make every item in the list click-able

```
listView.setOnItemClickListener(new OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent,
                            View view, int position, long id) {
        // When clicked, print out the text inside an item
        System.out.println(((TextView) view).getText())
    }
});
```


Android Application Development

Lesson 4

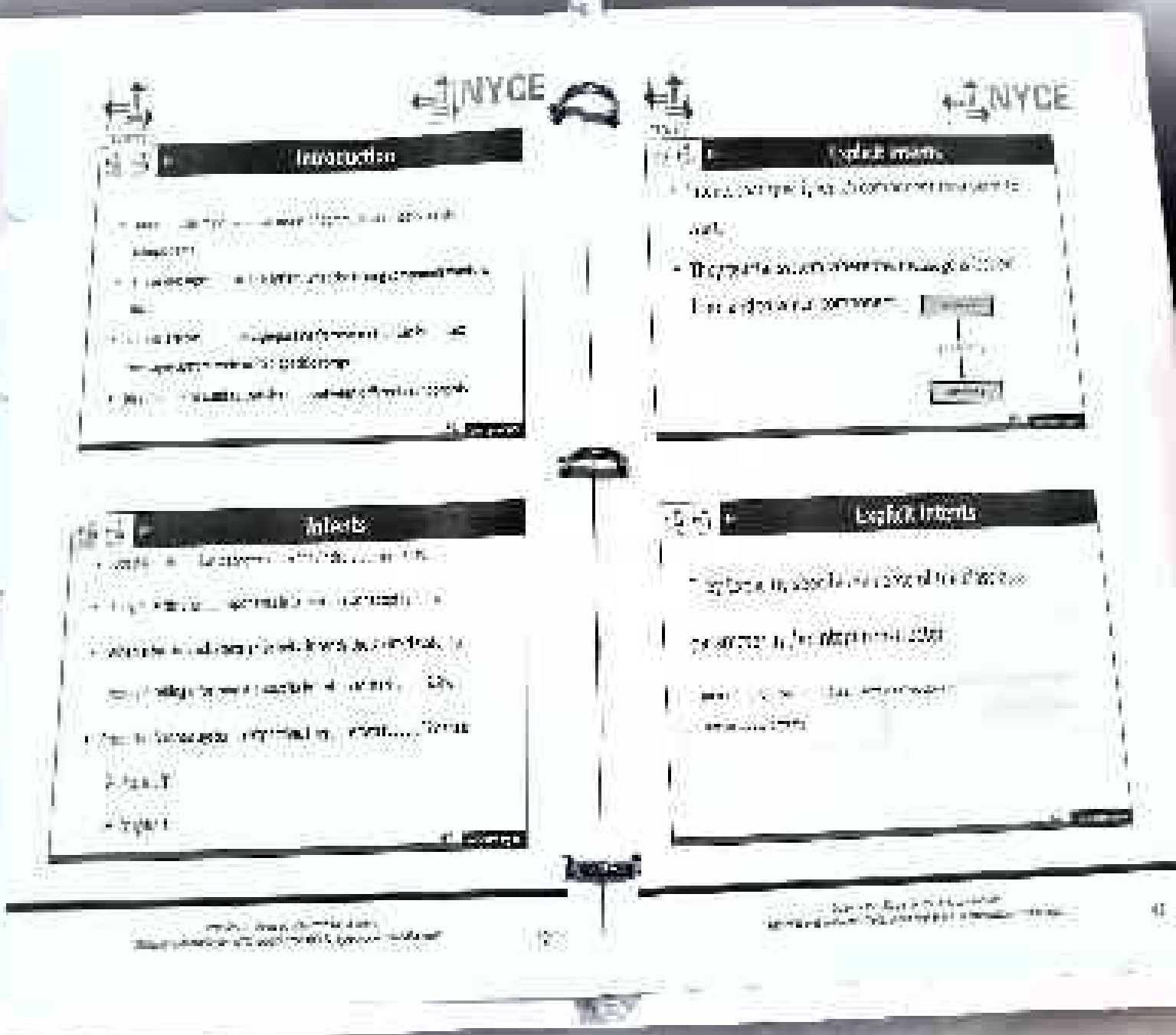
Intents and Intent filters

ATC

AND-401

Outline

- Introduction
- Intents
  - Explicit Intents
  - Implicit Intents
- Native Android Actions
- Data Transfer
- Intent to Call Activities
  - Direct calls
  - Sub-activities: Calling Activities for Results
- Register an IntentFilter
- Lab 4: Creating Contacts Selection Application



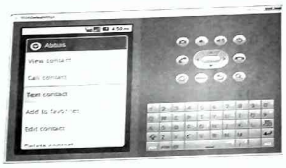
Implicit Intents

- Only specifies an Action
- The underlying Android system:
  - Translates the action.

```
Intent intent = new Intent(Intent.ACTION_VIEW,
    Uri.parse("http://www.androidatc.com"));
startActivity(intent);
```


Native Android Actions

- **ACTION_ANSWER**
  Opens an Activity that handles incoming calls.
- **ACTION_DELETE**
  Starts an Activity that allows you to delete the entry currently stored at the data URI location.
- **ACTION_DIAL**
  Brings up a dialer application with the number.



Native Android Actions

- **ACTION_EDIT**
  Requests an Activity that can edit the data at the URI.
- **ACTION_PICK**
  Launches a sub-Activity that allows you to pick an item from the URI data.
- **ACTION_VIEW**
  Asks that the data supplied in the Intent's URI be viewed in the most reasonable manner. It is the most common generic action.

Data Transfer

- Sending data:

```
Intent intent = new Intent(this, ActivityTwo.class);
intent.putExtra(Intent.EXTRA_TEXT, "News for you!");
```

- Receiving data:

```
Bundle extras = getIntent().getExtras();
if (extras == null) {
      return;
}
// Get data via the key
String value1 = extras.getString(Intent.EXTRA_TEXT);
if (value1 != null) {
      // Do something with the data
}
```

Register an IntentFilter

- When an implicit intent is sent, the Android system tries to match that intent's action and data with all activities that can process them.
- This process is simply a filtration of all activities registered in Android to receive implicit intents and is called intent resolution.

Register an IntentFilter

1. Modify Manifest file
   - Add `<intent-filter>` under the `<activity>` of the activity you want to register.
   - Add `<action>`, `<category>` and `<data>` under intent-filter tag

```
<activity android:name="MyListActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"> </action>
        <category android:name="android.intent.category.BROWSABLE">
        </category>
        <category android:name="android.intent.category.DEFAULT">
        </category>
        <data android:scheme="http"> </data>
    </intent-filter>
</activity>
```


Register an IntentFilter

2. Call getIntent().

```
Intent intent = getIntent();
if(intent.getAction().compareTo(Intent.ACTION_VIEW) == 0)
{
    Uri data = intent.getData();
    getListView().setFooterDividersEnabled(true);
    Toast.makeText(this, data.toString(), Toast.LENGTH_LONG).show();
    Log.v("log", intent.getAction());
    Log.v("log", Intent.ACTION_VIEW);
}
```

3. From another activity, make the implicit intent call.

```
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.androidatc.com"));
```


Android Application Development

Lesson 5
Layouts and Custom Views

ATC

AND-401

Outline

- Introduction
- Layouts
  - Layout Properties
  - Loading the Layout from Code
  - Creating and Editing Layouts in Android Studio
- Customized Views
- Modify Existing Views
  - Step by Step
  - What is in onDraw()?
- Lab 5: Custom View, Drawer Layout, and Fragments Application

95 www.nyce.org.mx

Introduction

- Each views has a default behavior an interface.
- Your application might require slightly different views that are not included in the built-in views.
- You might need to create your own views – custom views.

96 www.nyce.org.mx

Layouts

- **LinearLayout**. It arranges all widgets in a single column or row.
- **RelativeLayout**. It arranges widgets relative to each other.
- **TableLayout**. It arranges views into rows and columns
- **FrameLayout**. It allows all encapsulated widgets to be arranged on top of each other.

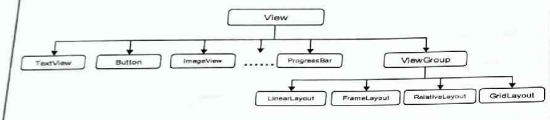
97 www.nyce.org.mx

Layouts

- **GridLayout**. It arranges views into a grid, splitting the screen into rows and columns. This is available since Android 4.0.
- **DrawerLayout.** This is a top-level container that allows having "drawer" views that can be pulled out from the edge of the window.

98 www.nyce.org.mx

Layouts



99 www.nyce.org.mx

Layout Properties

- *Each ViewGroup contains a nested class that extends class ViewGroup.LyoutParams.*
- *This nested class contains properties that define the size and position of each view inside it.*
- *These dimensions can be specified by an exact numerical value; or by one of the following two constants:*
  - ***MATCH_PARENT**: view will take the same size as the parent view.*
  - ***WRAP_CONTENT**: View will take the size of the content it is enclosing.*

100 www.nyce.org.mx

Layout Properties

frame_layout_sample.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.androidatc.fragments.ItemFragment">
    <ListView android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

101 www.nyce.org.mx

Layout Properties

Attribute	Description
android:id	The unique ID that identifies the view
android:layout_width	The width of the layout.
android:layout_height	The height of the layout
android:layout_marginTop	The space on the top side of the layout.
android:layout_marginBottom	The space on the bottom side of the layout.
android:layout_marginLeft	The space on the left side of the layout.
android:layout_marginRight	The space on the right side of the layout.
android:layout_gravity	Specifies how child Views are positioned
android:layout_weight	How much of the extra space in the layout should be allocated to the View
android:layout_x	This specifies the x-coordinate of the layout.
android:layout_y	This specifies the y-coordinate of the layout.

102 www.nyce.org.mx

Loading the Layout in Activity

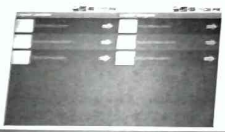
Call `setContentView()` in `onCreate()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.frame_layout_sample);
}
```


Loading the Layout in Activity

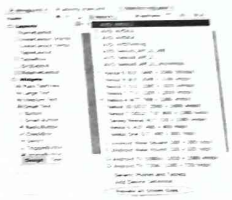
Call `inflator.inflate()` in `onCreateView()`

```
@Override
public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View rootView =
        inflater.inflate(R.layout.fragment_custom_view,
        container, false);
    return rootView;
}
```

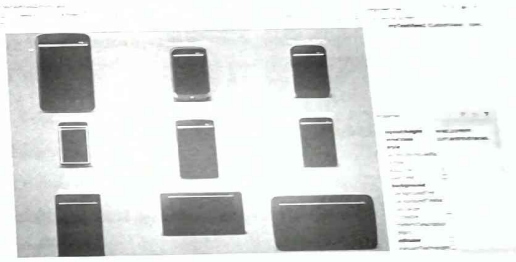


Creating and Editing Layouts in Android Studio

Preview the layout on multiple devices at once.



Creating and Editing Layouts in Android Studio



Customized Views

different types of customizations:

- ***Modify existing views:*** *You can create a different theme and/or behavior of an existing view provided by Android framework.*
- ***Compound Views:*** *Combine a group of views into a single view.*
- ***New views:*** *Create a totally new view that resembles real world object.*

Outline

- Introduction
- Android Resources
  - Why Using Resources?
  - Adding Resources to your Application
  - Using Resources
  - Types of Resources
- Android Themes and Styles
  - Creating Themes
- Android Material Design
  - Using the Material Theme
  - Color Palette
- Lab 6: A To-Do List Application in Material Design

Android Application Development

Lesson 6

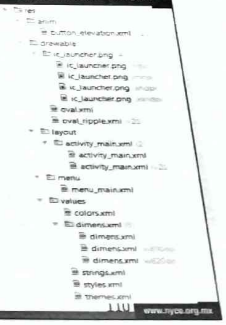
Android Resources, Themes, and Material Design

ATC

AND-401

Introduction

- Android application resources
  - How to include them in a package and how to use them.
- Adding styles and themes to your application activities
- The latest Android design guidelines called 'Material design'



Curso Desarrollo de aplicaciones con Android
Material elaborado para NYCE, queda prohibida la reproducción parcial o total.

- Building your Android application requires more than just writing Java code.
- You need to build the layout of your activities to provide the user interface.
- Layout files are xml files saved under */res/layout* and are a type many resource files you can include to your Android application.

111 www.nyce.org.mx

Why Using Resources?

- Whenever you need static and contact data (images, strings...etc.), you should add them as resources in your application.
- Improves your development experiences for several reasons:
  - Maintaining your application will be much easier, since your business code is separate from static interface code.
  - Debugging your application will be much easier since resources are independent from code.

112 www.nyce.org.mx

Why Using Resources?

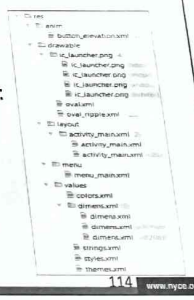
- You can reach much wider user base and cover much more Android-power devices by qualifying your resources.
- Improve the user interface and user experience.
- Easier development since you do not need to build every resource programmatically in your code.

113 www.nyce.org.mx

Adding Resources to your Application

Deciding which folder you should use to save a resource, depends on two factors:

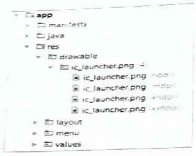
- The type of the resource.
- The device configuration that the resource serves.



114 www.nyce.org.mx

Using Resources

- Resources added in an application will receive a unique ID.
- The ID is a static integer variable that is compiled and built automatically by Android Studio.



Using Resources

After the ID is built by Android Studio, you can access it in Java code using the R class, or in XML using the @notation.

	Java code	XML
ID Format	R.resource_type.name	@resource_type/name
Example	R.string.app_name	@string/app_name

Types of Resources

Resource Type	Saved in	Reference	Description
View Animation	/res/anim	Java R.animator.filename XML @animator/filename	Resources that create an animation by modifying an object's property values over a set period of time using the Animator object.
Property Animation	/res/animator/	Java R.anim. filename XML @anim/filename	Resources for view animations. These include tween animations that perform a series of transformations on a single image, or frame animations which create an animation by showing a sequence of images.
Image files	/res/drawable	Java R.drawable.filename XML @drawable/filename	Resources that include any graphics-related files which can be drawn on the screen.
Launcher Icons	/res/mipmap	Java R.mipmap.filename XML @mipmap/filename	App launcher icons. Resources in this folder are retained by Android system regardless of the screen resolution of the device.

Types of Resources

Resource Type	Saved in	Reference	Description
Layout files	/res/layout	Java R.layout.filename XML @layout/filename	Resources that define the layout of your application to build the UI.
Strings	/res/values	Java R.string.string_name XML @string/string_name	Resources that define any set of string constants, or string arrays.
Menu Resources	/res/menu	Java R.menu.filename XML @menu.filename	Resources that define the content of your application's menu.
Styles	/res/values	XML @style/style_name Java R.color.color_name	Resources that define the look and feel of UI elements in the layout. Resources for all other types of constants you can include inside the values folder but have their own sub-classes in R file. (e.g. R.dimen, R.color...etc.).
Colors, Dimensions, Arrays	/res/values/	R.dimen.dimen_name R.array.array_name XML @color/color_name @dimen/dimen_name @array.array_name	

Android Themes and Styles

- Android SDK allows developers to create the style user interface of applications using XML resource files.
- Defining the style of user interfaces involves specifying values for the colors, fonts, dimensions...etc.
- When a resource is used to define the style for a view it is called a style.
- When it is used to style the whole activity or application it is called a theme.

Android Themes and Styles

- Styles and themes are defined and saved in XML files under */res* folder – specifically under ***/res/values/***.
- When you create a new project using Android Studio, a default style resource file called ***styles.xml*** is created under ***/res/values/***.

Creating Styles

- Define a style resource:

atc_view_styles.xml

```
<resources>
    <style name="atc_button_style">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:typeface">monospace</item>
        <item name="android:textColor">#00ff00ff</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textSize">20dp</item>
    </style>
</resources>
```

-

```
<Button
    android:text="Hello Styles!"
    android:id="@+id/button"
    style="@style/atc_button_style"/>
```


Creating Styles

- Apply style:

```
<Button
    android:text="Hello Styles!"
    android:id="@+id/button"
    style="@style/atc_button_style"/>
```

Creating Themes

Create a style that extends a system theme and apply it to the whole activity or application.

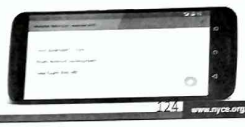
styles.xml

```
<resources>
  <style name="ATCTheme" parent="android:Theme.Material.Light">
    <item name="android:windowNoTitle">true</item>
    <item
      name="android:windowBackground">@color/background_material_light</item>
    <item name="android:button">@style/atc_button_style</item>
  </style>
</resources>
```

123 www.nyce.org.mx

Android Material Design

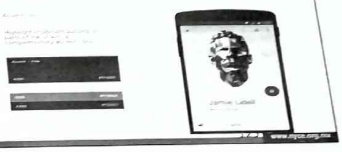
- It is one of the major updates that Android 5.0
- A comprehensive set of guides and rules for visual design for the UI.
- The material design borrows concepts of design from real life light, shadows, and movement of objects.
- Provides a user experience that is immersive and the functionality more apparent.



124 www.nyce.org.mx

Android Material Design

As of Android 5.0 Lollipop, the SDK provides a new theme and widgets and a set of APIs for shadows and animations.



Android Application Development

Lesson 7
Android UI – Dialogs, Menus, and WebView

ATC

AND-401

Outline

- Introduction
- User Interaction through Messages
- Dialogs
  - Dialog sub-classes
  - Creating dialogs with user-defined layout
  - Creating an Alert Dialog
  - Creating a Progress Dialog
  - Activities with Dialog Theme
- Toasts
- Menus
  - Sub-menus
  - Context Menus
  - Popup Menus
- WebView
- Lab 7: Wallpaper Application

127 www.nyce.org.mx

Introduction

Topics in this chapter:

- Usage of some essential Android UI components.
- Dialog component.
- Toast messages
- Menus.
- Web views.

128 www.nyce.org.mx

User Interaction through Messages

Interactions with users:

- Use Android's built-in Dialog sub-classes
- Create an Activity and make it behave as a Dialog by changing its theme.
- Toast: A special kind of message boxes that show up for a short period of time then disappear.

129 www.nyce.org.mx

Dialogs

- Small windows that appear in front of an activity.
- Occupy only part of the screen and can be partially transparent.
- The Activities behind them are either dimmed or blurred.

130 www.nyce.org.mx

Dialogs

- Part of an activity, and its life-cycle is controlled by the activity that is built in.
- No need to add it to the manifest.
- An important part of an Android application.
- Used to show messages to the user, show latest updates, take user's input ...etc.

131 www.nyce.org.mx

Dialog sub-classes

- **AlertDialog**

  A dialog that can manage zero, one, two, or three buttons, and/or a list of selectable items that can include checkboxes or radio buttons.

- **ProgressDialog**

  A dialog that displays a progress wheel or a progress bar. Because it's an extension of the AlertDialog, it also supports buttons.

- **DatePickerDialog**

  A dialog that allows the user to select a date.

- **TimePickerDialog**

  A dialog that allows the user to select a time.

132 www.nyce.org.mx

Custom Layout Dialogs

- Create layout resource, in the res/layout folder.
- Create a variable of type Dialog and instantiate it.
- Call `setContentView(R.Layout.dialog_view)` for the dialog
- Call `show()` method.

133 www.nyce.org.mx

Custom Layout Dialogs

```
final Dialog dialog = new Dialog(AndroidATCActivity.this);
dialog.setTitle("Dialog Title");
dialog.setContentView(R.layout.dialog_view);
TextView text = (TextView) dialog.findViewById(R.id.dialogBodyText);
text.setText("This is the text in my dialog");
dialog.show();
```

134 www.nyce.org.mx

AlertDialog

```
AlertDialog alertD = new AlertDialog.Builder(this).create();
alertD.setTitle("Back?");
alertD.setMessage("Are you sure you want to exit?");
alertD.setButton(AlertDialog.BUTTON_NEGATIVE, "NO", new OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
          dialog.cancel();
        }
});
alertD.show();
```

135 www.nyce.org.mx

Progress Dialog

```
Window window = dialog.getWindow();
ProgressDialog progressD = new ProgressDialog(this);
progressD.setProgressStyle(ProgressDialog.STYLE_SPINNER);
progressD.setButton(Dialog.BUTTON_NEGATIVE, "Cancel", new OnClickListener()
{
        @Override
        public void onClick(DialogInterface dialog, int which)
        {
        }
});
progressD.setMessage("Please wait. Loading...");
progressD.show();
```

136 www.nyce.org.mx

Toast

- An easy way to deliver status messages to users without disrupting the current working activity.
- Do not take focus away from the Activity.

137 www.nyce.org.mx

Toast

- Suitable for displaying informative messages that don't need too much attention.
- Fade away automatically after a certain period of time.
- It may not be guaranteed that a user will read the message.
- Are not used for critical messages.

```
Toast.makeText(this, "Your download has resumed.", Toast.LENGTH_LONG).show();
```

138 www.nyce.org.mx

Customized Toasts

```
Toast toast = Toast.makeText(this, " ",Toast.LENGTH_LONG);
toast.setGravity( Gravity.CENTER, 0, 0);

LinearLayout ll = new LinearLayout(context);
ll.setOrientation(LinearLayout.VERTICAL);
ll.setBackgroundColor(Color.DKGRAY);

TextView tv = new TextView(context);
tv.setText("Hello World From a Customized Toast");
Button btn = new Button(context);
btn.setText("   OK   ");
btn.setPadding(10, 10,10,10);

ll.addView(tv,LayoutParams.WRAP_CONTENT, LayoutParams.MATCH_PARENT);
ll.addView(btn, LayoutParams.WRAP_CONTENT,LayoutParams.MATCH_PARENT);
ll.setGravity(Gravity.CENTER);
toast.setView(ll);
toast.show();
```

Menus

- Provide user actions and other options in your activities.
- As of Android 3.0, menus are provided as part of an action bar of the activity.
- Each activity in an application can get its own **options menu**

Build an Options Menu

1. Create a new XML file under folder *res/menu*, if the file is not already available in the project.
2. Open the file in Android Studio.
3. Add the required menu items by adding the <item> tag for each menu item you want to create.

Build an Options Menu (Cont'd)

4. Add the required attributes for the <item> tag. Most importantly, fill in the item id and the title fields.

menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      tools:context=".MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never"/>
    <item
        android:id="@+id/action_exit"
        android:orderInCategory="101"
        android:title="@string/action_exit"
        app:showAsAction="never"/>
</menu>
```

Context Menu

1. Declare and initialize a View component, like `Button`, `TextView`, `EditText`... etc.
2. Register the view for context menu with `registerForContextMenu(view)` inside an activity's `onCreate()` method.
3. Override `onCreateContextMenu()` in the activity which takes three parameters.

143 www.nyce.org.mx

Context Menu

4. Call method `add()` of class `ContextMenu` which is similar to `add()` method of class Menu.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,          ContextMenu.ContextMenuInfo
menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);    menu.setHeaderTitle("Context Menu");
  menu.add(0, menu.FIRST, Menu.NONE, "Item 1").setIcon(R.drawable.menu_item);
  menu.add(0, menu.FIRST+1, Menu.NONE, "Item 2").setCheckable(true);
  menu.add(0, menu.FIRST+2, Menu.NONE, "Item 3").setShortcut('3', '3');
  SubMenu sub = menu.addSubMenu("Submenu");    sub.add("Submenu Item");
}
```

144 www.nyce.org.mx

Context Menu

145 www.nyce.org.mx

WebView

- You can embed a web browser inside your application using web views.
- A WebView is a crucial component for many applications that needs to display a website without worrying about using Android's native views.

146 www.nyce.org.mx

Android Application Development

Lesson 8
Android Storage and Background Processing

ATC

AND-401

www.nyce.org.mx

Outline

- Android Storage Options
- File I/O
- Shared Preferences
  - Retrieving Shared Preferences
  - Save activity state
- Connecting to the internet
- Background Processing
  - Java Threads and AsyncTask
  - Android Services
  - Background Fragments
- Lab 8: Quotes Provider Application

148 www.nyce.org.mx

Android Storage Options

- **Shared Preferences:** This technique stores application-specific primitive data in key-value pairs.
- **Internal Storage:** It stores private data on the device memory using file I/O
- **SQLite Databases:** They store structured data in a private database.
- **Network Connection:** It is used to store data on the web with your own network server.

149 www.nyce.org.mx

File I/O

Reading from local files and writing them in Android is purely Java-based. To write into a file, you should create a FileOutputStream instance, and to read from a file you should create a FileInputStream.

```
String FILE_NAME = "tempfile.tmp";
// Create a new output file stream that's private to this application.
FileOutputStream fos = openFileOutput(FILE_NAME, Context.MODE_PRIVATE);
// Create a new file input stream.
FileInputStream fis = openFileInput(FILE_NAME);
```

150 www.nyce.org.mx

Including files as resources

- Saved in a resource folder called raw.
- Packaged with application like other resources.
- To read a file as a resource:
  1. Create resource folder */res/raw*
  2. Add your file to the folder
  3. Access the file from the Java code using method

```
Resources myResources = getResources();
InputStream myFile = myResources.openRawResource(R.raw.myfilename);
```

151 www.nyce.org.mx

Shared Preferences

- Very useful and simple method to permanently store data.
- Recommended for storing primitive data.
- They are key-value pairs used for accessing and modifying preferences.
- Use `getSharedPreferences()` to get an instance of Shared Preferences object.
- Use `SharedPreferences.Editor` object to modify preferences.

152 www.nyce.org.mx

Shared Preferences

The following code uses `SharedPreferences` to save a string.

```
public static final String MY_PREFS = "mySharedPreferences";
protected void savePreferences(){
    // Create or retrieve the shared preference object.
    int mode = Activity.MODE_PRIVATE;
    SharedPreferences mySharedPreferences = getSharedPreferences(MY_PREFS, mode);
    // Retrieve an editor to modify the shared preferences.
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    // Store new primitive types in the shared preferences object.
    editor.putString("textEntryValue", "Not Empty");
    // Commit the changes.
    editor.commit();
}
```

153

Retrieving Shared Preferences

You need to know the key of the shared preference and the type of value stored i.e. whether it's string, int, boolean,...etc.

```
public void loadPreferences() {
  // Get the stored preferences
  int mode = Activity.MODE_PRIVATE;
  SharedPreferences mySharedPreferences =
                    getSharedPreferences(MYPREFS,mode);
  // Retrieve the saved values.
  String stringPreference
     = mySharedPreferences.getString("textEntryValue","");
}
```

154 www.nyce.org.mx

Save activity state

- One of the best scenarios to use shared preferences is saving the UI state of an activity.
- Suppose an activity in your application contains a set of form widgets, and after filling the form out partially, the leaves the application.
- Whenever the user resumes an activity, you would load the preferences you have saved when the activity was paused.

Save Activity State

Save an activity state in four simple steps:

1. Create a method to save the state of an activity.
2. Create another method to load the saved activity state.
3. Override the Activity's onPause( ) call-back to call method in step 1.
4. Override the Activity's onResume( ) call-back to and call method in step 2.

Connecting to the internet

Follow these steps:

1. Connect to a URL that retrieves data you can process.
2. Retrieve the data from the URL.
3. Process and manipulate retrieved data.

Connecting to the internet

```
private String connectToWeb() {
  String url = "http://www.AndroidATC.com";
  DefaultHttpClient client = new DefaultHttpClient();
    HttpGet getRequest = new HttpGet(url);
      try {
        HttpResponse getResponse = client.execute(getRequest);
        final int statusCode = getResponse.getStatusLine().getStatusCode();
        if (statusCode != HttpStatus.SC_OK) {
              Log.w(getClass().getSimpleName(), "Error for URL" + url);
                                    return null;
        }
        HttpEntity getResponseEntity = getResponse.getEntity();
        byte[] buffer = new byte[128];
        getResponseEntity.getContent().read(buffer);
        return new String(buffer);
      }
      catch (IOException e) {
           getRequest.abort();
      }
  return null;
}
```

Background Processing

- Android threads.
- Android Services.
- Background fragments

159 www.nyce.org.mx

Android Threading Model

- ***UI thread***
  - *Also called main thread. This is basically the Activity thread and it is created when you launch an application.*
- ***Worker thread.***
  - *Any thread spawned from a UI thread to perform some task in the background.*

160 www.nyce.org.mx

Android Threading Model

- Two rules to Android's single thread model that you must always follow when designing your application:
  - Do not block the UI thread
  - Do not access the Android UI from outside the UI thread
- To create worker threads from inside your UI thread, you can either use Java threads or use Android's AsyncTask.

161 www.nyce.org.mx

AsyncTask

- Allows asynchronous processing on your UI.
- Creates a worker thread and sends results back to UI thread.
- Does not require the developer to handle threads and handlers.
- Ideal for short background operations.

152 www.nyce.org.mx

Steps to Use AsyncTask

1. Create a subclass of AsynTask.

```
public class GetTemperatureTask extends AsyncTask<String, Void, String>{ … }
```

2. Override method doInBackground(), connect to the internet inside it, and return the data retrieved.

```
public class GetTemperatureTask extends AsyncTask<String, Void, String>{
        @Override
        protected Boolean doInBackground(String... params) {
            String url = params[0];
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);
            // Connect to the URL and save retrieved data in dataResponse.
            return dataResponse;
        }
}
```

163 www.nyce.org.mx

Steps to Use AsyncTask (Cont'd)

3. Override other AysncTask methods to perform an UI changes.

```
public class GetTemperatureTask extends AsyncTask<String, Void, String>{
String dateResponse = null;
            @Override
       protected void onPreExecute() {
           super.onPreExecute();
           temperatureTextView.setText("Loading...");
       }
       @Override
       protected Boolean doInBackground(String... params) { … }
       @Override
       protected void onPostExecute(String result) {
           super.onPostExecute(result);
           dateTimeTextView.setText(dateResponse);
    }
}
```


Steps to Use AsyncTask (Cont'd)

4. Create an instance of your AsynTask subclass in your activity and call its execute() method.

```
protected void onCreate(Bundle savedInstanceState) {
  …
  new GetTemperatureTask().execute(url);
  …
}
```

165 www.nyce.org.mx

AsyncTask

```
public class GetTemperatureTask extends AsyncTask<String, Void, String>
```

Parameters sent to the task upon execution.

Progress units published during the background processing.

Result of the background processing.

166 www.nyce.org.mx

Android Services

- Performs long running processes - without UI - to handle networking operations, streaming, updating widgets...etc.
- Has a lifecycle that is independent from the component that started it (activity)
- Runs on the main thread, so any intensive processing should be done on a thread created inside the service.

167 www.nyce.org.mx

Background Fragments

- Fragments without a view can also be used as a background worker to perform your background processing.
- When you inflate such fragments, you set the view to null.
- Like any fragment the lifecycle of this fragment is tied to that of the Activity.

168 www.nyce.org.mx

Android Application Development

Lesson 9

Android Storage: SQLite and Content Providers

ATC

AND-401

Outline

- Introduction
- SQLite Database in your application
- SQLite library
- SQLiteDatabase
- Cursors
- Databases in Android
- Content Providers
  - Native Android Content Providers
  - Custom Content Provider
  - Designing the Storage for Data
  - Steps to Create a Content Provider
- Sync Adapters
- Lab 9: SQLite Databases and Content Providers

170 www.nyce.org.mx

Introduction

- SQLite is an Open Source light-weight Database system embedded into every Android device.
- No database setup or administration for using a SQLite database in Android.
- You only have to define the SQL statements for creating and updating the database.
- Database is automatically managed Android.

171 www.nyce.org.mx

SQLite Database in your application

- Databases are saved by default in directory

/DATA/data/application_name/databases/DATABASE_FILE.

- application_name is your application's name.
- DATABASE_FILE is the DB name you specify in code.
- Database names should be unique in your application.

172 www.nyce.org.mx

SQLite library

- Using SQLite library in your application requires the following:
  - Importing package **android.database.***
  - Usage of the classes `SQLiteOpenHelper`, `SQLiteDatabase`, `Cursor`, and `SQLiteOpenHelper`
- `SQLiteOpenHelper` is a helper class to manage database creation and version management.
- It is recommended that you create a subclass of `SQLiteOpenHelper` to perform the open, create, or upgrade operation of the database.

173 www.nyce.org.mx

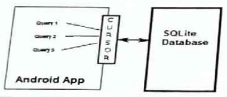
SQLiteDatabase

SQLiteDatabase exposes methods to manage a SQLite database. It has methods to create, delete, execute SQL commands, and perform other common database management tasks.

174 www.nyce.org.mx

Cursors

- Provides random read-write access to the result set returned by a database query.
- It is a pointer to the data returned by a database query.
- It doesn't contain all the data from your query, but rather used to facilitate access.



175 www.nyce.org.mx

Steps to use Databases in Android

1. Create an adapter class
2. Create an inner private class that extends

   `SQLiteOpenHelper`
3. Create the class constructor.

176 www.nyce.org.mx

Steps to use Databases in Android

4. Create inner class constructor.
5. Override `onCreate()` of inner class.
6. Override `onUpgrade()`

177 www.nyce.org.mx

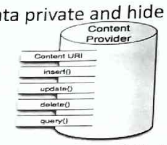
Steps to use Databases in Android

7. Create method `open()`
8. Create DB operations
   - Delete
   - Update
   - Select
   - Insert

178 www.nyce.org.mx

Content Providers

- *One of the building blocks of Android applications.*
- *Manage access to a central repository of data.*
- *Android applications keep the data private and hide it from other applications.*



179 www.nyce.org.mx

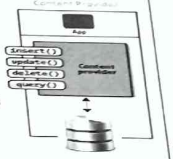
Content Providers

- *Content providers are mainly used to share the data across other applications.*
- *Encapsulate the data and provide it to other applications through a single content resolver interface.*

180 www.nyce.org.mx

Native Android Content Providers

- Android allows you to access some of its system resources using Content Provider
- You can, query all the contacts in phone using the contacts URI provided by Android's People class.



181 www.nyce.org.mx

Native Android Content Providers

Receive the name and phone number of contacts in your device:

```
// Get a cursor over every contact.
Cursor cursor = getContentResolver().query(People.CONTENT_URI,
null, null, null, null);
// Let the activity manage the cursor lifecycle.
startManagingCursor(cursor);
// Use the convenience properties to get the index of the columns
int nameIdx = cursor.getColumnIndexOrThrow(People.NAME);
int phoneIdx = cursor.getColumnIndexOrThrow(People.NUMBER);
String[] result = new String[cursor.getCount()];
```

182 www.nyce.org.mx

Custom Content Provider

- To share the data of your application with other applications, you can create own content provider.
- Common cases that might require creating a content provider:
  - Complex data and files need to be offered to other applications.
  - Allow the users to copy some complex data from your application.
  - Provide custom search suggestions using the search framework of Android

183

Designing the Storage for Data

- Data can be provided by content providers in two ways.
- **File Data**

  Any file like video, image or an audio.

- **Structured Data**

  Any database, arrays etc. in this case the application stores the data that is compatible with tables of rows and columns.

184

Android Application Development

Lesson 10
Android Notifications

ATC

AND-401

Outline

- Introduction
- Creating a notification
- Notification actions
- Steps to Create a Simple Notification
- Expandable Notifications
- Notifications Layouts
- Notification Priority
- Notifications in Android 5.0 (Lollipop)
  - Lock Screen Notifications
  - Heads-up Notifications
- Lab 10: Implementing Android Notifications

186

Introduction

- Notification Manager service is used to handle notifications from different applications.
- You can create a new notification for certain events in your application.
- A small icon appears for each application notification in the notification bar.

187 www.nyce.org.mx

Creating a notification

To create notifications, you should use the `NotificationManager` class which can be created in an Activity using the `getSystemService()` method.

```
NotificationManager MyNotificationManager
= (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

188 www.nyce.org.mx

Notification actions

- A notification action allows users to go directly from the notification to an Activity in your application.
- You can add more actions by adding more UI elements (like a button) to the notification.
- Inside a Notification, a `PendingIntent` defines the action itself.

189 www.nyce.org.mx

Notification Actions

Here is how to create a notification and an explicit Intent to be used by a `PendingIntent`.

```
// Create Notifcation
Notification notification = new Notification(R.drawable.icon,  "Hello notifications!",
System.currentTimeMillis());

// Cancel the notification after its selected
notification.flags |= Notification.FLAG_AUTO_CANCEL;
notification.number += 1;

// Specify the called Activity
Intent intent = new Intent(this, NotificationReceiver.class);

PendingIntent activity = PendingIntent.getActivity(this, 0, intent, 0);
notification.setLatestEventInfo(this, "This is the title",
                                "This is the text", activity);
notificationManager.notify(0, notification);
```

190 www.nyce.org.mx

Steps to Create a Simple Notification

1. Create a NotificationManager object.
2. Create notification object
3. Configure appearance
4. Trigger the notification
5. You can cancel the notification after triggering it.

Expandable Notifications

- Introduced in Android 4.1 Jelly Bean.
- They allow defining big views in notifications that are displayed when the notification is expanded.
- Following are the three styles that can be used in expandable notifications.
  - Big picture style
  - Big text style
  - Inbox style

Big Picture Style

- When dragged down, it shows a bigger image that is set before sending the notification.
- To create this type of notifications:

```
Notification.Builder builder = new Notification.Builder(this)
.setStyle()(
   new Notification.BigPictureStyle().bigPicture(image)
   );
```

Big Text Style

- When dragged down, it reveals more text details.
- Implemented as the big Picture with just a difference in using methods:
  - `bigText("")`
  - `setBigContentTitle("")`

Inbox Style

- Mostly used with messages received.
- When the notification is dragged down, the text in the received message appears in the expanded view of the notification.
- To implement the inbox style notification the object of `NotificationCompat.InboxStyle` is made like this:

```
NotificationCompat.InboxStyle inboxStyle = new
NotificationCompat.InboxStyle();
```

195 www.nyce.org.mx

Notifications Layouts - Base Layout

- All the notifications consist of base layout.
- Contains the following information
  - The sender's photo or sending application's notification icon
  - Notification title and a message
  - A timestamp.
  - The secondary icon to identify the sending application.

196 www.nyce.org.mx

Notifications Layouts – Expanded Layout

- Used in expanded notification.
- Can contain the bigger images or text message.

197 www.nyce.org.mx

Notification Priority

Priority	Use
MAX	Used for critical and urgent notifications.
HIGH	Used high priority notifications primarily for important communication.
DEFAULT	The default priority.
LOW	notifications you still want the user to be informed about, but they rate low in urgency.
MIN	Contextual/background information . Minimum priority notifications will not show in the status bar.

198 www.nyce.org.mx

TECNYCE

NYCE

Notifications in Android 5.0 (Lollipop)

- Visual changes consistent with the new material design theme.
- Available on the device lock screen, while hiding sensitive content.
- High-priority notifications can use a new heads-up notifications format.
- Cloud-synced notifications.

Lock Screen Notifications

- Appear on top of screen even if locked.
- Media playback applications use them.
- Android uses sorting algorithms to prioritize them.
- These algorithms take into account:
  - The timestamp and application's stated priority.
  - Whether the notification has recently disturbed the user.
  - Any people attached to the notification.

TECNYCE

NYCE

Heads-up Notifications

- High, max or full-screen priority notifications turn into heads-up notifications.
- These are presented to the users for a short period of time.
- Example of heads-up notifications can be
  - Incoming phone call when using a device
  - Alarm when using a device
  - New SMS message
  - Low battery